

EV436703024

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Search Engine With Natural Language-Based Robust
Parsing of User Query and Relevance Feedback
Learning**

Inventor(s):
Hai-Feng Wang
Kai-Fu Lee
Qiang Yang

ATTORNEY'S DOCKET NO. MS1-606US

TECHNICAL FIELD

This invention relates to search engines and other information retrieval tools.

BACKGROUND

With the explosive growth of information on the World Wide Web, there is an acute need for search engine technology to keep pace with users' need for searching speed and precision. Today's popular search engines, such as "Yahoo!" and "MSN.com", are used by millions of users each day to find information. Unfortunately, the basic search method has remained essentially the same as the first search engine introduced years ago.

Search engines have undergone two main evolutions. The first evolution produced keyword-based search engines. The majority of search engines on the Web today (e.g., Yahoo! and MSN.com) rely mainly on keyword searching. These engines accept a keyword-based query from a user and search in one or more index databases. For instance, a user interested in Chinese restaurants in Seattle may type in "Seattle, Chinese, Restaurants" or a short phrase "Chinese restaurants in Seattle".

Keyword-based search engines interpret the user query by focusing only on identifiable keywords (e.g., "restaurant", "Chinese", and "Seattle"). Because of its simplicity, the keyword-based search engines can produce unsatisfactory search results, often returning many irrelevant documents (e.g., documents on the Seattle area or restaurants in general). In some cases, the engines return millions of documents in response to a simple keyword query, which often makes it impossible for a user to find the needed information.

1 This poor performance is primarily attributable to the ineffectiveness of
2 simple keywords being capable of capturing and understanding complex search
3 semantics a user wishes to express in the query. Keyword-based search engines
4 simply interpret the user query without ascribing any intelligence to the form and
5 expression entered by the user.

6 In response to this problem of keyword-based engines, a second generation
7 of search engines evolved to go beyond simple keywords. The second-generation
8 search engines attempt to characterize the user's query in terms of predefined
9 frequently asked questions (FAQs), which are manually indexed from user logs
10 along with corresponding answers. One key characteristic of FAQ searches is that
11 they take advantage of the fact that commonly asked questions are much fewer
12 than total number of questions, and thus can be manually entered. By using user
13 logs, they can compute which questions are most commonly asked. With these
14 search engines, one level of indirection is added by asking the user to confirm one
15 or more rephrased questions in order to find an answer. A prime example of a
16 FAQ-based search engine is the engine employed at the Web site
17 "Askjeeves.com".

18 Continuing our example to locate a Chinese restaurant in Seattle, suppose a
19 user at the "Askjeeves.com" site enters the following search query:

20
21 "What Chinese restaurants are in Seattle?"
22

23 In response to this query, the search engine at the site rephrases the
24 question as one or more FAQs, as follows:
25

1 How can I find a restaurant in Seattle?

2 How can I find a yellow pages listing for restaurants in Seattle, WA?

3 Where can I find tourist information for Seattle?

4 Where can I find geographical resources from Britannica.com on Seattle?

5 Where can I find the official Web site for the city of Seattle?

6 How can I book a hotel in Seattle?

7
8 If any of these rephrased questions accurately reflect the user's intention,
9 the user is asked to confirm the rephrased question to continue the searching
10 process. Results from the confirmed question are then presented.

11 An advantage of this style of interaction and cataloging is much higher
12 precision. Whereas the keyword-based search engines might return thousands of
13 results, the FAQ-based search engine often yields a few very precise results as
14 answers. It is plausible that this style of FAQ-based search engines will enjoy
15 remarkable success in limited domain applications, such as web-based technical
16 support.

17 However, the FAQ-based search engines are also limited in their
18 understanding the user's query, because they only look up frequently occurring
19 words in the query, and do not perform any deeper syntactic or semantic analysis.
20 In the above example, the search engine still experiences difficulty locating
21 "*Chinese* restaurants", as exemplified by the omission of the modifier "*Chinese*"
22 in any of the rephrased questions. While FAQ-based second-generation search
23 engines have improved search precision, there remains a need for further
24 improvement in search engines.
25

1 Another problem with existing search engines is that most people are
2 dissatisfied with the user interface (UI). The chief complaint is that the UI is not
3 designed to allow people to express their intention. Users often browse the
4 Internet with the desire to obtain useful information. For the keywords-based
5 search engine, there are mainly two problems that hinder the discovery of user
6 intention. First, it is not so easy for users to express their intention by simple
7 keywords. Second, keyword-based search engines often return too many results
8 unrelated to the users' intention. For example, a user may want to get travel
9 information about Beijing. Entering 'travel' as a keyword query in Yahoo, for
10 example, a user is given 289 categories and 17925 sites and the travel information
11 about Beijing is nowhere in the first 100 items.

12 Existing FAQ-based search engines offer UIs that allow entry of pseudo
13 natural language queries to search for information. However, the underlying
14 engine does not try to understand the semantics of the query or users' intention.
15 Indeed, the user's intention and the actual query are sometimes different.

16 Accordingly, there is a further need to improve the user interface of search
17 engines to better capture the user's intention as a way to provide higher quality
18 search results.

19 20 SUMMARY

21 A search engine architecture is designed to handle a full range of user
22 queries, from complex sentence-based queries to simple keyword searches. The
23 search engine architecture includes a natural language parser that parses a user
24 query and extracts syntactic and semantic information. The parser is robust in the
25 sense that it not only returns fully-parsed results (e.g., a parse tree), but is also

1 capable of returning partially-parsed fragments in those cases where more accurate
2 or descriptive information in the user query is unavailable. This is particularly
3 beneficial in comparison to previous efforts that utilized full parsers (i.e., not
4 robust parsers) in information retrieval. Whereas full parsers tended to fail on
5 many reasonable sentences that were not strictly grammatical, the search engine
6 architecture described herein always returns the best fully-parsed or partially-
7 parsed interpretation possible.

8 The search engine architecture has a question matcher to match the fully-
9 parsed output and the partially-parsed fragments to a set of frequently asked
10 questions (FAQs) stored in a database. The question matcher correlates the
11 questions with a group of possible answers arranged in standard templates that
12 represent possible solutions to the user query.

13 The search engine architecture also has a keyword searcher to locate other
14 possible answers by searching on any keywords returned from the parser. The
15 search engine may be configured to search content in databases or on the Web to
16 return possible answers.

17 The search engine architecture includes a user interface to facilitate entry of
18 a natural language query and to present the answers returned from the question
19 matcher and the keyword searcher. The user is asked to confirm which answer
20 best represents his/her intentions when entering the initial search query.

21 The search engine architecture logs the queries, the answers returned to the
22 user, and the user's confirmation feedback in a log database. The search engine
23 has a log analyzer to evaluate the log database and glean information that
24 improves performance of the search engine over time. For instance, the search
25 engine uses the log data to train the parser and the question matcher. As part of

1 this training, the log analyzer is able to derive various weighting factors indicating
2 how relevant a question is to a parsed concept returned from the parser, or how
3 relevant a particular answer is to a particular question. These weighting factors
4 help the search engine obtain results that are more likely to be what the user
5 intended based on the user's query.

6 In this manner, depending upon the intelligence provided in the query, the
7 search engine's ability to identify relevant answers can be statistically measured in
8 terms of a confidence rating. Generally, the confidence ratings of an accurate and
9 precise search improve with the ability to parse the user query. Search results
10 based on a fully-parsed output typically garner the highest confidence rating
11 because the search engine uses essentially most of the information in the user
12 query to discern the user's search intention. Search results based on a partially-
13 parsed fragment typically receive a comparatively moderate confidence rating,
14 while search results based on keyword searching are given the lowest confidence
15 rating.

16 17 **BRIEF DESCRIPTION OF THE DRAWINGS**

18 Fig. 1 is a block diagram of an exemplary computer network in which a
19 server computer implements a search engine for handling client queries.

20 Fig. 2 is a block diagram of a search engine architecture.

21 Fig. 3 is a flow diagram of a search process using the search engine.

22 Fig. 4 is a block diagram of a robust parser employed in the search engine.

23 Fig. 5 is a diagrammatic illustration of a tokenization of a Chinese sentence
24 to demonstrate the added difficulties of parsing languages other than English.
25

1 Fig. 6 is a flow diagram of a question matching process employed in the
2 search engine.

3 Fig. 7 illustrates database tables used during the question matching process
4 of Fig. 6.

5 Fig. 8 illustrates a first screen view of Chinese-version search engine user
6 interface implemented by the search engine.

7 Fig. 9 illustrates a second screen view of Chinese-version search engine
8 user interface implemented by the search engine.

9 10 **DETAILED DESCRIPTION**

11 This disclosure describes a search engine architecture that handles a full
12 range of user queries, from complex sentence-based queries to simple keyword
13 searches. Unlike traditional search engines, the architecture includes a natural
14 language parser that parses a user query and extracts syntactic and semantic
15 information. The parser is *robust* in the sense that it not only returns fully-parsed
16 results, but is also capable of returning partially-parsed fragments in those cases
17 where more accurate or descriptive information in the user query is unavailable.

18 When facing ambiguity, the search engine architecture interacts with the
19 user for confirmation in terms of the concept the user is asking. The query logs
20 are recorded and processed repeatedly, thus providing a powerful language model
21 for the natural language parser as well as for indexing the frequently asked
22 questions and providing relevance-feedback learning capability.

23 The search engine architecture is described in the context of an Internet-
24 based system in which a client submits user queries to a server and the server hosts
25 the search engine to conduct the search on behalf of the client. Moreover, the

1 search engine architecture is described as handling English and Chinese
2 languages. However, the architecture may be implemented in other environments
3 and extended to other languages. For instance, the architecture may be
4 implemented on a proprietary local area network and configured to handle one or
5 more other languages (e.g., Japanese, French, German, etc.).

6 7 **Exemplary Computing Environment**

8 Fig. 1 shows an exemplary computer network system 100 in which the
9 search engine architecture may be implemented. The network system 100
10 includes a client computer 102 that submits user queries to a server computer 104
11 via a network 106, such as the Internet. While the search engine architecture can
12 be implemented using other networks (e.g., a wide area network or local area
13 network) and should not be limited to the Internet, the architecture will be
14 described in the context of the Internet as one suitable implementation.

15 The client 102 is representative of many diverse computer systems,
16 including general-purpose computers (e.g., desktop computer, laptop computer,
17 etc.), network appliances (e.g., set-top box (STB), game console, etc.), and
18 wireless communication devices (e.g., cellular phones, personal digital assistants
19 (PDAs), pagers, or other devices capable of receiving and/or sending wireless data
20 communication). The client 102 includes a processor 110, a volatile memory 112
21 (e.g., RAM), a non-volatile memory 114 (e.g., ROM, Flash, hard disk, optical,
22 etc.), one or more input devices 116 (e.g., keyboard, keypad, mouse, remote
23 control, stylus, microphone, etc.) and one or more output devices 118 (e.g.,
24 display, audio speakers, etc.).

1 The client 102 is equipped with a browser 120, which is stored in non-
2 volatile memory 114 and executed on processor 110. The browser 120 facilitates
3 communication with the server 104 via the network 106. For discussion purposes,
4 the browser 120 may be configured as a conventional Internet browser that is
5 capable of receiving and rendering documents written in a markup language, such
6 as HTML (hypertext markup language).

7 In the illustrated implementation, the server 104 implements a search
8 engine architecture that is capable of receiving user queries from the client 102,
9 parsing the queries to obtain complete phrases, partial phrases, or keywords, and
10 returning the appropriate results. The server 104 is representative of many
11 different server environments, including a server for a local area network or wide
12 area network, a backend for such a server, or a Web server. In this latter
13 environment of a Web server, the server 104 may be implemented as one or more
14 computers that are configured with server software to host a site on the Internet
15 106, such as a Web site for searching.

16 The server 104 has a processor 130, volatile memory 132 (e.g., RAM), and
17 non-volatile memory 134 (e.g., ROM, Flash, hard disk, optical, RAID memory,
18 etc.). The server 104 runs an operating system 136 and a search engine 140. For
19 purposes of illustration, operating system 136 and search engine 142 are illustrated
20 as discrete blocks stored in the non-volatile memory 134, although it is recognized
21 that such programs and components reside at various times in different storage
22 components of the server 104 and are executed by the processor 130. Generally,
23 these software components are stored in non-volatile memory 134 and from there,
24 are loaded at least partially into the volatile main memory 132 for execution on the
25 processor 130.

1 The search engine 140 includes a robust parser 142 to parse a query using
2 natural language parsing. Depending on the search query, the robust parser
3 produces a fully-parsed output (e.g., a parse tree), one or more partially-parsed
4 fragments, and/or one or more keywords. A FAQ matcher 144 matches the fully-
5 parsed output (e.g., a parse tree) and the partially-parsed fragments to a set of
6 possible frequently asked questions that are stored in a database. The FAQ
7 matcher then correlates the questions with a group of possible answers to the user
8 query. A keyword searcher 146 attempts to locate other possible answers from
9 conducting keyword searching using the keywords returned from the parser.

10 Unlike traditional engines, the search engine architecture robustly
11 accommodates many types of user queries, from single keyword strings to full,
12 grammatically correct sentences. If the user enters a complete sentence, the search
13 engine 140 has the ability to parse the sentence for syntactic and semantic
14 information. This information better reveals the user's intention and allows for a
15 more precise search with higher quality results. If the user enters a grammatically
16 incorrect sentence or an incomplete sentence (i.e., a phrase), the search engine 140
17 attempts to map the partial fragments to FAQ concepts. Finally, even if the user
18 query contains only one or a few search terms, the search engine is able to handle
19 the query as a keyword-based search and return at least some results, albeit not
20 with the same precision and quality.

21 The search engine 140 presents the possible answers returned from the
22 FAQ matcher 144 and the keyword searcher 146 to a user. The user is asked to
23 confirm which of the answers best represents the user's intentions in the query.
24 Through this feedback, the search engine may refine the search. Additionally, the
25

1 search engine may use this relevance feedback to train the architecture in its
2 mapping of a parsed query into relevant answers.

3 The search engine includes a query log analyzer 148 that tracks the query,
4 the returned results, and the user's feedback to those results in a log database. The
5 query log analyzer 148 analyzes the log database to train the FAQ matcher 144.
6 As part of this training, the query log analyzer 148 is able to derive, over time,
7 various weights indicating how relevant a FAQ is to a parsed concept generated by
8 parsing a particular query, or how relevant a particular answer is to a particular
9 FAQ. These weights help the search engine obtain results that are more likely to
10 be what the user intended based on the user's query.

11 In this manner, depending upon the intelligence provided in the query, the
12 search engine's ability to identify relevant answers can be statistically measured in
13 terms of a confidence rating. Generally, the confidence ratings of an accurate and
14 precise search improve with the ability to parse the user query. Search results
15 based on a fully-parsed output typically garner the highest confidence rating
16 because the search engine uses essentially most of the information in the user
17 query to discern the user's search intention. Search results based on a partially-
18 parsed fragment typically receive a comparatively moderate confidence rating,
19 while search results based on keyword searching are given the lowest confidence
20 rating.

21 22 Search Engine Architecture

23 The search engine architecture 140 is formulated according to an
24 underlying premise, referred to as the *concept-space hypothesis*, that a *small*
25 *subset of concepts cover most user queries*. Examples of concepts are: "Finding

1 computer and internet related products and services ", "Finding movies and toys
2 on the Internet", and so on. It is believed that the first few popular categories will
3 actually cover most of the queries. Upon analyzing a one-day log from MSN.com,
4 the inventors discovered that 30% of the concepts covered approximately 80% of
5 all queries in the selected query pool.

6 Fig. 2 illustrates the search engine architecture 140 in more detail. It has a
7 search engine user interface (UI) 200 that seamlessly integrates search
8 functionality and browsing. In the Fig. 1 network system, the search engine UI
9 200 is served in an HTML document to the client 102 when the client initially
10 addresses the Web site. One exemplary implementation of the user interface 200
11 is described below in more detail beneath the heading "Search Engine User
12 Interface".

13 The user enters a search query via the search engine UI 200. A query string
14 is passed to the natural language-based robust parser 142, which performs robust
15 parsing and extracts syntactic as well as semantic information for natural language
16 queries. The robust parser 142 includes a natural language parser (NLP) 202 that
17 parses the query string according to rules kept in a rules database 204. The parsed
18 output is ranked with a confidence rating to indicate how likely the output
19 represents the query intentions.

20 The output of the natural language robust parser 142 is a collection of
21 concepts and keywords. The concepts are obtained through a semantic analysis
22 and include a fully-parsed output (e.g., a parse tree) and partially-parsed
23 fragments. One suitable semantic analysis is described below in the section under
24 the heading "NL-based Robust Parsing". The keywords are either the key phrases
25

1 extracted directly from the user query or are expanded queries through a synonym
2 table.

3 After natural language processing, the concepts and keywords are passed on
4 to the FAQ matcher 144. The FAQ matcher 144 has a FAQ matching component
5 206 that attempts to match the concepts and keywords to predefined frequently
6 asked questions stored in a FAQ database 208. From the FAQs, the FAQ
7 matching component 206 identifies related templates from a template database 210
8 that group together similar question parameters. The templates have associated
9 indexed answers that are maintained in an answer database 212.

10 Accordingly, the FAQ matcher 144 effectively maps the parsed concepts
11 and keywords to FAQs, the FAQs to templates, and the templates to answers. In
12 one implementation, the FAQ database 208 is configured as a relational database
13 that maintains a set of tables to correlate the concepts, FAQs, templates, and
14 answers. One example database structure is described below with reference to
15 Fig. 7.

16 Concurrent with FAQ-based searching, the NLP module 142 also sends the
17 keywords to a keyword-based module 146 for keyword searching on the user's
18 query. The keyword-based module 146 has a meta-search engine 214 that
19 extracts answers from the Web 216.

20 The answers returned from the FAQ matcher 144 and keyword searcher
21 146 are presented to the user via UI 200. The user is asked to confirm which, if
22 any, of the returned answers best exemplifies the user's intentions in the query.
23 By analyzing which results the user selects, the search engine may further refine
24 the search using the confirmed answer as a starting point and return even more
25 accurate results.

1 In addition to facilitating various search levels in an integrated manner, the
2 search engine architecture 140 also supports a query log analyzer 148 that
3 implements methodology to process query logs for the purpose of obtaining new
4 question templates with indexed answers. It also has relevance-feedback
5 capability for improving its indexing and ranking functions. This capability
6 allows the architecture 140 to record users' actions in browsing and selecting the
7 search result, so that the ranking of these results and the importance of each
8 selection can be learned over time.

9 The architecture has a log collector 218 to log user actions and system
10 output in a log database 220. Log data mining tools 222 may be used to analyze
11 the log database 220 to glean data used to refine the FAQ database 208, template
12 database 210, answer database 212, and FAQ matching functions 206. A web
13 crawler 224 may also be included to provide information as needed from the Web
14 216.

15 In one implementation, the search engine architecture 140 may be
16 configured according to COM (Component Object Model) or DCOM (Distributed
17 COM). This allows for design modularity, allowing each individual module to
18 evolve independently from others as long as the inter-module interface remains the
19 same.

20 Compared to the traditional search engines, the search engine architecture
21 140 offers many benefits, including a higher precision and search efficiency on
22 frequently asked questions. Additionally, the indexed contents evolve with users'
23 current interests and its ranking ability improves with usage over time. The search
24 engine architecture scales easily to offer relatively large coverage for user's
25

1 questions and the natural user interface allow users to seamlessly integrate search
2 and browsing.

3 4 Search Process

5 Fig. 3 shows a search process 300 conducted on the search engine
6 architecture 140 of Fig. 2. The search process 300 is implemented as computer
7 executable instructions that, when executed, perform the operations illustrated as
8 blocks in Fig. 3. Selected operations of the search process 300 are described after
9 this section in more detail.

10 At block 302, the search engine 140 receives a user query entered at remote
11 client 102. At block 304, the user query is parsed at the natural language robust
12 parser 142 to produce the parsed concepts (if any) and keywords. After parsing,
13 the concepts and keywords are submitted to the FAQ matcher 144 to match them
14 with frequently asked questions in the FAQ database (block 306). Upon
15 identifying matched FAQs, the FAQ matcher 144 identifies associated templates
16 with indexed answers from databases 210 and 212 to obtain answers for the user
17 queries (block 308).

18 Concurrent to the FAQ-matching operations, the search engine also
19 performs a keyword search at keyword-based module 146 (block 310). At block
20 312, the results of the FAQ matching and keyword searching are presented to the
21 user via the search engine UI 200. The user is then given the opportunity to offer
22 feedback in an attempt to confirm the accuracy of the search.

23 Meanwhile, apart from the search functions, the search engine is also
24 providing relevance feedback learning through analysis of the query, the returned
25 results and the user feedback to the search results. At block 314, the log collector

1 218 logs user queries, results returned to the user, and selections made by the user.
2 These records are stored in the log database 220.

3 At block 316, the log database 220 is analyzed to ascertain frequently asked
4 questions from a large number of user questions and to automatically develop or
5 find answers for the questions. The log is further analyzed to determine weights
6 indicating how probable the returned results pertain to the users' queries (block
7 318). In particular, the log analyzer determines how likely the FAQs represent the
8 user queries and how likely the answers pertain to the FAQs. The weightings are
9 used to modify the FAQ matcher 144 (block 320).

10 11 **NL-Based Robust Parsing (Block 304)**

12 The natural language-based robust parser 142 employs robust parsing to
13 accommodate many diverse types of user queries, including full and partial
14 sentences, meaningful phrases, and independent search terms. User queries are
15 often entered into search engines as incomplete or grammatically incorrect
16 sentences. For instance, users who want to know about Chinese restaurants in
17 Seattle might enter queries quite differently, as illustrated by the following
18 examples:

19
20 Chinese restaurants in Seattle

21 Seattle's best Chinese restaurants

22 Any Chinese restaurants in Seattle?

23 Where is the closest Chinese restaurant?

24 What is the best Chinese restaurant in Seattle?
25

1 While it is difficult to parse such sentences using a traditional natural
2 language parser, the robust parser 142 is capable of handling such partial or
3 grammatically incorrect sentences. Unlike traditional parsing that require a
4 hypothesis and a partial parse to cover adjacent words in the input, robust parsing
5 relaxes this requirement, making it possible to omit noisy words in the input. If a
6 user query contains words that are not parsable, the natural language parsing
7 module 142 can skip these words or phrases and still output a result.

8 Additionally, different hypotheses can result from partial parses by
9 skipping some symbols in parse rules. Thus, if a given sentence is incomplete
10 such that natural language parsing is unable to find a suitable rule to match it
11 exactly, the robust parser provides multiple interpretations of the parsing result
12 and associates with each output a confidence level. In the search engine 140, this
13 confidence level is built based on statistical training.

14 Fig. 4 shows an exemplary implementation of the natural language robust
15 parser 142. The module includes a word segmentation unit 400, which identifies
16 individual words in a sentence. The word segmentation unit 400 relies on data
17 from a query log 402 and a dictionary 404. In English, words are separated by
18 spaces and hence, word segmentation is easily accomplished. However, in other
19 languages, segmentation is not a trivial task. With Chinese text, for example, there
20 is no separator between words. A sequence of characters may have many possible
21 parses in the word-tokenization stage. Thus, effective information retrieval of
22 Chinese first requires good word segmentation.

23 Fig. 5 shows an example tokenization 500 of a simple Chinese sentence
24 “马上下来”, having only four characters. Here, these four characters can be
25 parsed in five ways into words. For example, the dotted path 502 represents a

1 parsing to the phrase “*dismounted a horse*”, and the bold path 504 represents
2 “*immediately coming down*”. This figure also shows seven possible “words”,
3 some of which (e.g., 上下) might be disputable on whether they should be
4 considered “words.”

5 To accommodate Chinese input, the robust parser can accept two kinds of
6 input: Lattice and N-best. The lattice input includes almost all possible
7 segmentations. However, as there may be too much ambiguity, the parsing
8 process can become very slow. An alternative choice is to use the N-best input.

9 With reference again to Fig. 4, after segmentation, the segmented sentence
10 is passed a natural language parser ⁴¹²410 and a keyword module ⁴¹⁰412. The parser
11 ⁴¹²410 attempts to parse the segmented sentence according to a set of rules found in a
12 rule database ⁴¹⁶414. If a sentence parses successfully, the parsing module ⁴¹²412
13 outputs a parse tree. If parsing is unsuccessful, the keyword unit ⁴¹⁰412 uses a word
14 database ⁴¹⁴416 to extract and output keywords from the segmented sentence. As
15 shown in Fig. 2, the parse tree and keywords are passed to the FAQ matcher 144
16 and the keywords are passed to the keyword-based component 146. Accordingly,
17 the architecture 140 allows templates to be matched regardless of the type of
18 output, whether parse trees or keywords.

20 Exemplary Parsing Methodology

21 One particular implementation of a robust parser is based on a spoken
22 language system known as “LEAP”, which stands for Language Enabled
23 Applications. LEAP is technology being developed in Microsoft Research that
24 aims at spoken language understanding. For a more detailed discussion of LEAP,
25 the reader is directed to an article by Y. Wang, entitled “A robust parser for

1 spoken language understanding", *Proc. of 6th European conference on speech*
2 *communication and technology (Eurospeech99)*, Budapest, Hungary, Sept. 1999,
3 pp. Vol.5, 2055-2058.

4 The robust parser employs a parsing algorithm that is an extension of a
5 bottom-up chart-parsing algorithm. The grammar defines *semantic classes*. Each
6 semantic class is defined by a set of rules and productions. For example, a
7 semantic class <Route> is defined for the travel path from one place to another.
8 This class is represented as follows:

```
10      <Route> TravelPath {  
11          =>      @from      <PlaceName:place1>      @to  
12          <PlaceName:place2> @route;  
13          @from => from | ...;  
14          .....  
15      }  
16      <PlaceName> Place {  
17          Beijing | Shanghai | ...;  
18      }
```

19
20 In the semantic classes above, <Route> defines a return class type, and
21 TravelPath is a semantic class that contains a number of rules (the first line) and
22 productions (the second line). In this class, "@from" parses a piece of the input
23 sentence according to a production as shown in the second line. The input item
24 after the "@from" object matches according to a <PlaceName> semantic class. If
25 there are input tokens that are not parsable by any parts of the rule, it will be

1 ignored by the parser. In this case, the scoring of the parse result will be
2 correspondingly discounted to reflect a lower level of confidence in the parse
3 result:

4 As an example, suppose the input query is:

5
6 从北京到上海应该怎么走 ? (How to go from Beijing
7 to Shanghai?)
8

9 The robust parser will return the following result:

10
11 <VOID> 从 place 到 place 怎么走
12 <Route> 从 place 到 place 怎么走
13 <PlaceName:place1> place
14 <PlaceName:place2> place
15

16 Here <VOID> represents the root semantic class. Note that this input query
17 cannot be parsed using the first rule in the semantic class TravelPath if a
18 traditional parser is used because the Chinese word “应该” cannot match any
19 objects in the rule. Since the robust parser can skip this word to match the rest,
20 parsing will continue to produce a partial result. In one implementation, the score
21 of the parsing result is calculated by discounting the number of input items and
22 rule items that are skipped during the parsing operation. This score is normalized
23 to give a percentage confidence value.
24
25

Evaluating Parsing Results

A parsed result will be selected if it covers the most words in the query and the most parts of rules. To improve the scoring strategy, the search engine learns probabilities from query logs, including:

- probabilities of the rules;
- penalty for robust rule matching (insertion, deletion, substitution);
- probabilities of “non-matching” words;
- term probability according to their frequency in query log.

Considering the rule in the semantic class <Route> TravelPath:

```
@from <PlaceName:place1> @to <PlaceName:place2>  
@route;
```

The search engine can train the probabilities associated with this rule. A rule with a high probability value means that using the rule to parse a query is more reliable. The search engine can also train the penalty values for robust matching by exacting a penalty for any item in either a rule or the query sentence that is skipped during parsing.

Consider the above rule for the sentence “从北京到上海应该怎么走” (“How to get from Beijing to Shanghai?”). A relatively low penalty is set if the @from item “应该(should)” is skipped. A higher penalty is assigned if the @route item “怎么走(how to go)” is skipped.

1 Statistics are gathered using the query log files as the base data. A more
2 detailed discussion of training the robust parser using query log files is described
3 below beneath the heading "Training Robust Parser Using Query Log Files".
4

5 Question Matching (Blocks 306 and 308)

6 The FAQ matcher 144 attempts to find a set of relevant concepts and their
7 related answers from a given user query. To accomplish this, the FAQ matcher
8 144 maps the concepts through several intermediate spaces to ultimately identify
9 answers to the queries.

10 Fig. 6 shows a mapping process 600 of the question matching operation.
11 The mapping process 600 is implemented as computer executable instructions
12 that, when executed, perform the operations illustrated as blocks in Fig. 6. For
13 discussion purposes, the mapping process is described in the context of a realistic
14 example in which a user asks:

15
16 从北京到上海应该怎么走 ? ("How to go from
17 Beijing to Shanghai?")
18

19 At block 602, the FAQ matcher maps the parsed query from a query space
20 to a concept or FAQ space. The natural language processing module 142 returns a
21 parse tree containing a semantic class and its parameters:

22 <VOID> 从 place 到 place 怎么走
23 <Route> 从 place 到 place 怎么走
24 <PlaceName:place1> place
25 <PlaceName:place2> place

1
2 A collection of concepts indexed on "交通" ("Route") and "旅游"
3 ("Travel"), and possibly other related concepts, are stored in the FAQ database
4 208.

5 Fig. 7 illustrates example database tables 700 maintained in the FAQ
6 database 208. In this example, the FAQ database is configured as a relational
7 database in which data records are organized in tables that may be associated with
8 one another using definable relationships. The database includes a Concept-FAQ
9 table 702, a FAQ table 704, a template table 706, and an answer table 708. For
10 this example, the answer table 708 pertains to answers about a flight schedule, and
11 hence is labeled as a "Flight Table".

12 The Concept-FAQ table 702 is the core data structure for the whole
13 database. It correlates concepts with frequently asked questions (FAQs). A FAQ
14 is made up of a few concepts that are in fact represented by certain terms, such as
15 "Route". Every FAQ is related to one or more concepts and every concept is
16 related to one or more FAQs. Thus, there is a many-to-many relationship between
17 FAQs and concepts. Every FAQ is assigned a FAQ ID to uniquely distinguish
18 FAQs from one another.

19 A record in the Concept-FAQ table 702 includes a concept, a FAQ ID, and
20 a weight. Each record indicates that a FAQ (with a particular ID) is related to the
21 concept according to a correlation weighting factor. The weighting factor
22 indicates how probable the concept pertains to the associated FAQ. The weighting
23 factor is learned from a later analysis of the query log file.
24
25

Using the Concept-FAQ table 702, the FAQ matcher 144 computes a correlation between a concept set $\Phi(\text{concept}_1, \text{concept}_2, \dots, \text{concept}_n)$ and a FAQ with ID of x as follows:

$$\sum_{i=1}^n \text{Weight}(\text{concept}_i, x).$$

Hence, given a concept set, the FAQ matcher can obtain the top n best-matched FAQs. For example, the concept set of the question “从北京到上海应该怎么走” (“How to go from Beijing to Shanghai”) are “Travel” and “Route”, where the match result is a FAQ set{101(weight 165), 105(weight 90)}.

The semantic class returned from the parser is used to search the concept-FAQ table. In our example, the semantic class “Route” is used as a key to search the Concept-FAQ table 702. The search determines that the third entry 710 in the table yields a perfect match. Corresponding to the “route” entry 710 is the FAQ with ID “101”, which can be used to index the FAQ table 704.

At block 604 in the mapping process of Fig. 6, the FAQ matcher maps the FAQs from the FAQ space to a template space. A template represents a class of standard questions and corresponds to a semantic class in the robust parser. Every template has one or more parameters with values. Once all the parameters in a template are assigned a value, a standard question is derived from this template.

For example, “*有哪些航班 (Which flights are there)” is a template representing a class of questions about the flight from or to a certain location. Here, the wild card “*” denotes that there is a parameter in the template that can be assigned an arbitrary place name. If “上海(Shanghai)” is chosen, this template

1 is transformed into a standard question “上海有那些航班(Which Shanghai flights
2 are there)”.

3 The FAQ table 704 associates frequently asked questions with templates.
4 The FAQ table 704 may also include a weight to indicate how likely a FAQ
5 pertains to a template. In our example, the frequently asked question with an ID
6 of “101” has three entries in the FAQ table 704, identifying three corresponding
7 templates with IDs 18, 21 and 24. Template 24 carries a weight of “100”,
8 indicating that this template is perhaps a better fit for the given FAQ than the other
9 templates. The template IDs can then be used to index into the template table 706.

10 The template table 706 correlates template IDs with template descriptions
11 and identities of corresponding answer sets. In Fig. 7, for example, the template
12 with ID 18 corresponds to an answer table that is named “Flight Table.”

13 It is infeasible to construct a template for every question because there are
14 many similar questions. Instead a single template is prepared for all similar
15 questions. This effectively compresses the FAQ set. In our example, the mapping
16 result for FAQ set {101, 105} is a template set {24(*的列车时刻表, weight
17 165+100), 18(*的航班有哪些, weight 165+80), 21(*的轮船班次情况, weight
18 165+50), 31(*有什么著名的旅游景点, weight 90+75)}, where the weights are
19 obtained by a simple addition of the weights from previous steps.

20 At block 606 in the mapping process of Fig. 6, the FAQ matcher maps
21 templates from the template space to an answer space. All answers for a
22 template are previously stored in a separate answer table, such as answer table
23 708. The answer table is indexed by parameter values of the template. When
24 matching is done, the best parameter is calculated and passed to the search
25 engine UI 200 to be shown to the user.

As shown in answer table 708, every answer has two parts: a URL and its description. In our example, if the user chooses a template 18 (*的航班有哪些), and value of the parameter is assigned to "上海", the flight table is returned with the portion of "上海" in the table shown to the user.

Training Robust Parser Using Query Log Files

The search engine architecture 140 uses information mined by the log analyzer 148 to adapt the robust parser 142 so that it evaluates the output based on the coverage of a rule against the input query. A parsed result will be selected if it covers the most words in the query and the most parts of rules. To improve the scoring strategy, probabilities learned from query logs include:

- confidence values associated with each rule;
- confidence values associated with each item in a rule;
- confidence values associated with each word in an input sentence.

First, consider the confidence values associated with each rule. To evaluate the parsing result more accurately, each rule is assigned a probability. Since the rules are local to a semantic class, the sum of probabilities of all the rules in a semantic class is one. Considering a semantic class having n rules, the probabilities of the i^{th} rule is w_i , then

$$\sum_i w_i = 1$$

1 The productions in grammar are either global or local to a semantic
2 class. The probabilities for all *global productions* (the productions always
3 available) that expand the same item sum to one. The probabilities for all
4 *productions local to one semantic class* (the productions only available within a
5 semantic class) that expand the same item sum to one too.

6 After learning the probabilities for each rule, the next task is to learn the
7 confidence values associated with each item in a rule. Considering a rule
8 having N items, robust matching is performed on the rule. Suppose the items
9 $T_{i_1}, T_{i_2}, \dots, T_{i_n}$ are matched, but the items $T_{j_1}, T_{j_2}, \dots, T_{j_n}$ ($1 \leq i, j \leq N$) are not matched.
10 A confidence value indicating how well this rule is matched is then measured.
11 The measurement may be performed, for example, by using neural networks.

12 One suitable implementation is to use a perceptron to measure the
13 confidence. A perceptron has N input units, each of them representing an item
14 in the rule, and one output unit, which represents the confidence of the rule
15 matching. To represent the confidence continually, which is not Boolean, a
16 Sigmoid function is used as the activation function for the output unit. For the
17 matched item T_{i_1} , the corresponding input is $I_{i_1} = C_{i_1}$, in which C_{i_1} is the
18 confidence of I_{i_1} ; whereas for the non-matched item T_{j_1} , the input is $I_{j_1} = 0$.

19 The output unit is:

$$c_r = \text{sigmoid}(\sum_p w_{rp} I_p)$$

22 where w_{rp} is the weight from input unit I_p to output unit. A standard gradient
23 descent method is used to train the perceptron, such as that described in S.
24 Russell, P. Norvig, "Artificial Intelligence", Prentice-Hall, Inc. 1995, pp573-
25

1 577. The training data is the user query log file where the sentences are
2 classified as positive and negative examples.

3 Finally, after learning the confidence values associated with each item in a
4 rule, the last task is to learn the confidence values associated with each word in an
5 input sentence. A non-matching word is the word in the input sentence that does
6 not match any item in the rule. For a word w , if there are n non-matching
7 occurrence in the training corpus, and if $m(m \leq n)$ of them result in correct rule-
8 matching, then the confidence of this non-matching is: $p = m/n$. The confidence
9 of the robust sentence matching is:

$$11 \quad c_s = \prod_i p_i$$

12
13 The confidence of a rule r is calculated as below:

$$14 \quad P = w_r \cdot c_r \cdot c_s$$

16 Search Engine User Interface

17 The search engine UI 200 is designed to improve efficiency and accuracy in
18 information retrieval based on a user's search intention. The intention-centric UI
19 design guides users to a small number of high-quality results, often consisting of
20 fewer than ten intention-related answers. The "intention" of a search on the
21 Internet is a process rather than an event. The search engine UI 200 attempts to
22 capture the process as three main tasks. First, users are permitted to pose queries
23 as natural language questions. Second, the UI presents parameterized search
24
25

1 results from the search engine and asks users to confirm their intention. Finally,
2 users are permitted to select their desired answer.

3 Fig. 8 shows an example screen display 800 of the search engine UI 200.
4 The screen display has a query entry area 802 that allows user to enter natural
5 language questions. Consider, for example, the following two queries in the
6 traveling domain search:

7
8 从（北京）到（上海）的旅游路线有哪些? (How many
9 traveling routes exist from (Beijing) to (Shanghai)?)

10
11 （北京）有哪些著名的景点? (Please tell me about the famous
12 sights in (Beijing)?)

13
14 Natural language is a powerful tool for expressing the user intention. The
15 most important parts of a query are referred to as *core phrases*. In these examples,
16 the underlined words are core phrases, the parenthesized words are keywords, and
17 the remaining words are redundant words.

18 In some cases, it is difficult or impossible to identify users' intention from
19 the original query alone. In this case, the search engine selects all possibly
20 relevant concept templates and asks the user to confirm. Related concepts are
21 clustered according to their similarity and the different parts of the result are
22 treated as parameters. From the above query, two similar search results
23 “北京的著名景点” (“famous sites in Beijing”) and “; 上海著名景点” (“famous
24 sites in Shanghai”) are combined into one group, where 北京 (Beijing) and
25 上海(Shanghai) are treated as parameters.

1 Fig. 9 shows an exemplary display screen 900 that is returned with various
2 parameterized search results. The result “(北京|上海) 著名景点” (famous
3 sites in [Beijing|Shanghai]) is depicted in result area 902. The parameterized
4 result can help focus users’ attention on the core phrases, which in this case
5 corresponds to “著名景点” (famous sites).

6 In addition to intention centricity, the search engine UI is designed to
7 seamlessly integrate searching and browsing. The search engine UI is constructed
8 with a strong sense of structure and navigation support so that users know where
9 they are, where they have been, and where they can go. In particular, there are
10 two kinds of combination modes for search and browsing: (1) browsing followed
11 by searching, and (2) searching followed by browsing.

12 For discussion purposes, suppose a user wants to know how to travel to
13 Shanghai for fun. At first, the user does not know what kind of information the
14 web can provide. The user can open a travel information-related web site and find
15 that there is information about “travel routes” (旅游路线). At this point, the user
16 may pose a query about the specific route to go to Shanghai from Beijing by
17 asking, for example, “从北京到上海旅游路线有哪些?” (“How to get from
18 Beijing to Shanghai?”)

19 Alternatively, the user may wish to search first, rather than browse to a
20 travel web site. After the user inputs a natural language query, the search engine
21 judges the user intention by using the core phrases. Because the intention extends
22 beyond a simple question, the search engine predicts the user’s intention from the
23 current query and provides reasonable answers for confirmation. For example, in
24 the above example, the real goal of the user is to get useful information about
25 traveling to Shanghai. Thus, the sightseeing information about Shanghai is related

1 to the user's intention. In response to the above query, the search results are two
2 alternative answers related to the user's intention:

- 3
- 4 • 从（北京）到（上海）的旅游路线; . (The sightseeing routes from
- 5 Beijing to Shanghai)
- 6 • （上海）的旅游景点. (The sightseeing sites in Shanghai)
- 7

8 Conclusion

9 A new-generation search engine for Internet searching permits natural
10 language understanding, FAQ template database matching and user interface
11 components. The architecture is configured to precisely index frequently asked
12 concepts and intentions from user queries, based on parsed results and/or
13 keywords.

14 Although the description above uses language that is specific to structural
15 features and/or methodological acts, it is to be understood that the invention
16 defined in the appended claims is not limited to the specific features or acts
17 described. Rather, the specific features and acts are disclosed as exemplary forms
18 of implementing the invention.

19

20

21

22

23

24

25